Poland, Gdańsk
September 9-12, 2007
19°
51°
Seventh International Conference on Parallel
Processing and Applied Mathematics
PPAM 2007

Future Directions in High Performance Computing

Jack Dongarra

INNOVATIVE COMPUTING LABORATORY

University of Tennessee
Oak Ridge National Laboratory
The University of Manchester

Outline

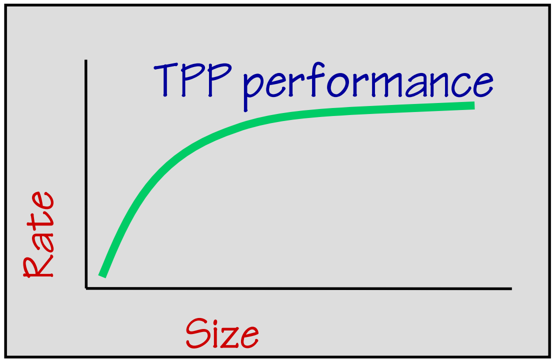
- **Top500 Results**
- **Four Important Concepts that Will Effect Math Software**
  - **Effective Use of Many-Core**
  - **Exploiting Mixed Precision in Our Numerical Computations**
  - **Self Adapting / Auto Tuning of Software**
  - **Fault Tolerant Algorithms**

TOP500 superCOMPUTER

H. Meuer, H. Simon, E. Strohmaier, & JD

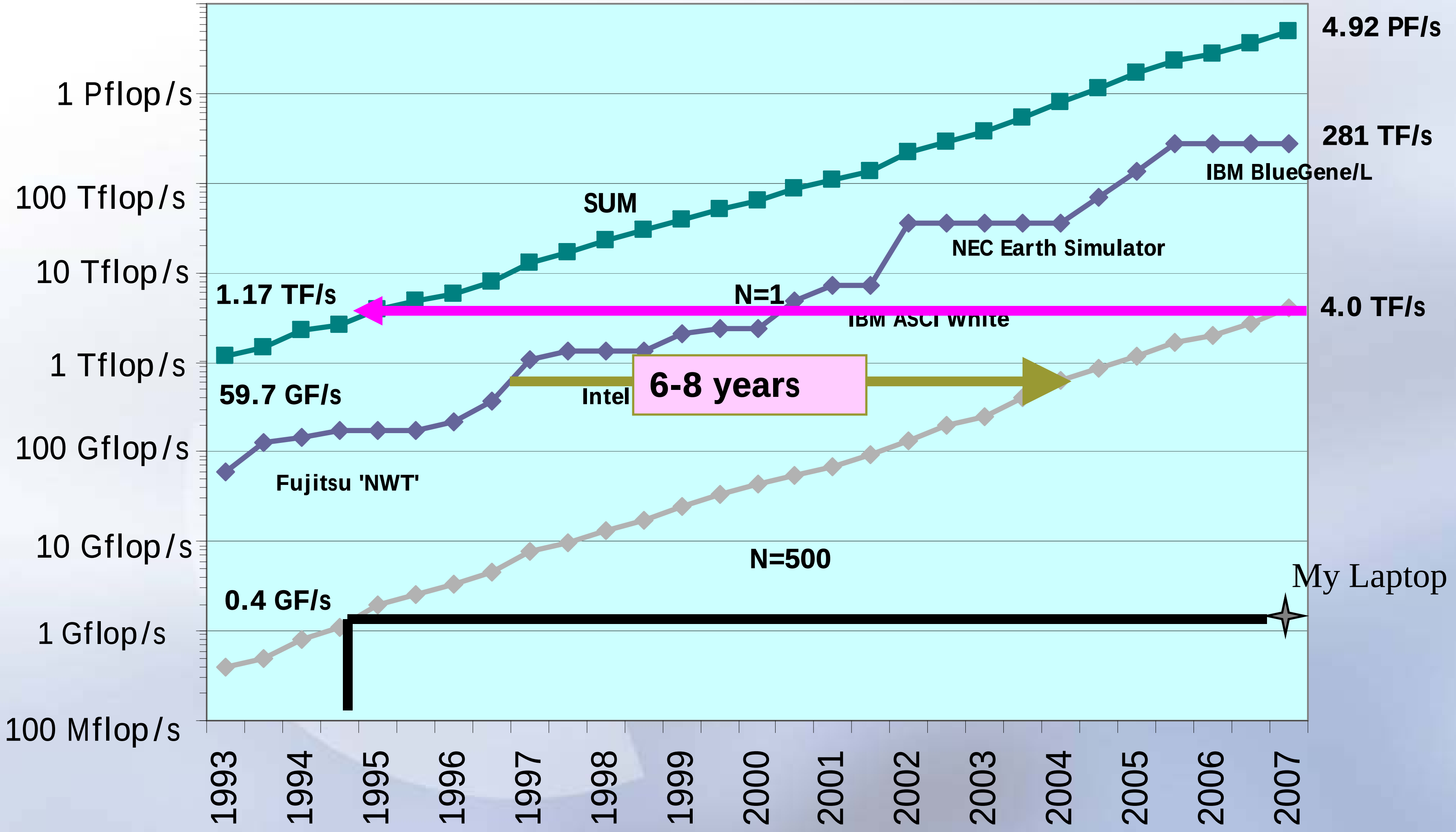
- Listing of the 500 most powerful Computers in the World
- Yardstick: Rmax from LINPACK MPP

  Ax=b, dense problem



- Updated twice a year
  SC‘xy in the States in November
  Meeting in Germany in June
- All data available from **www.top500.org**

Performance Development

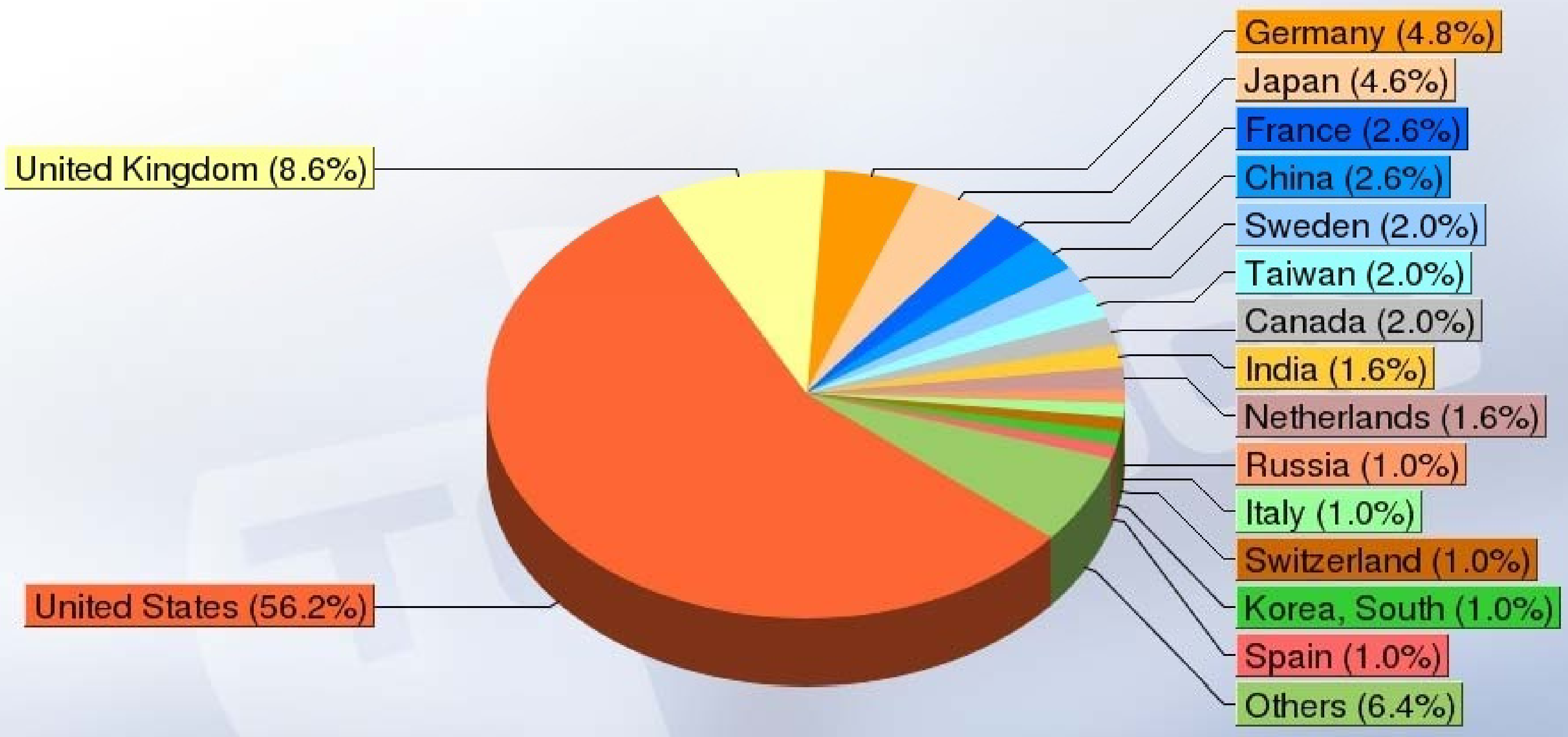


29th List: The TOP10

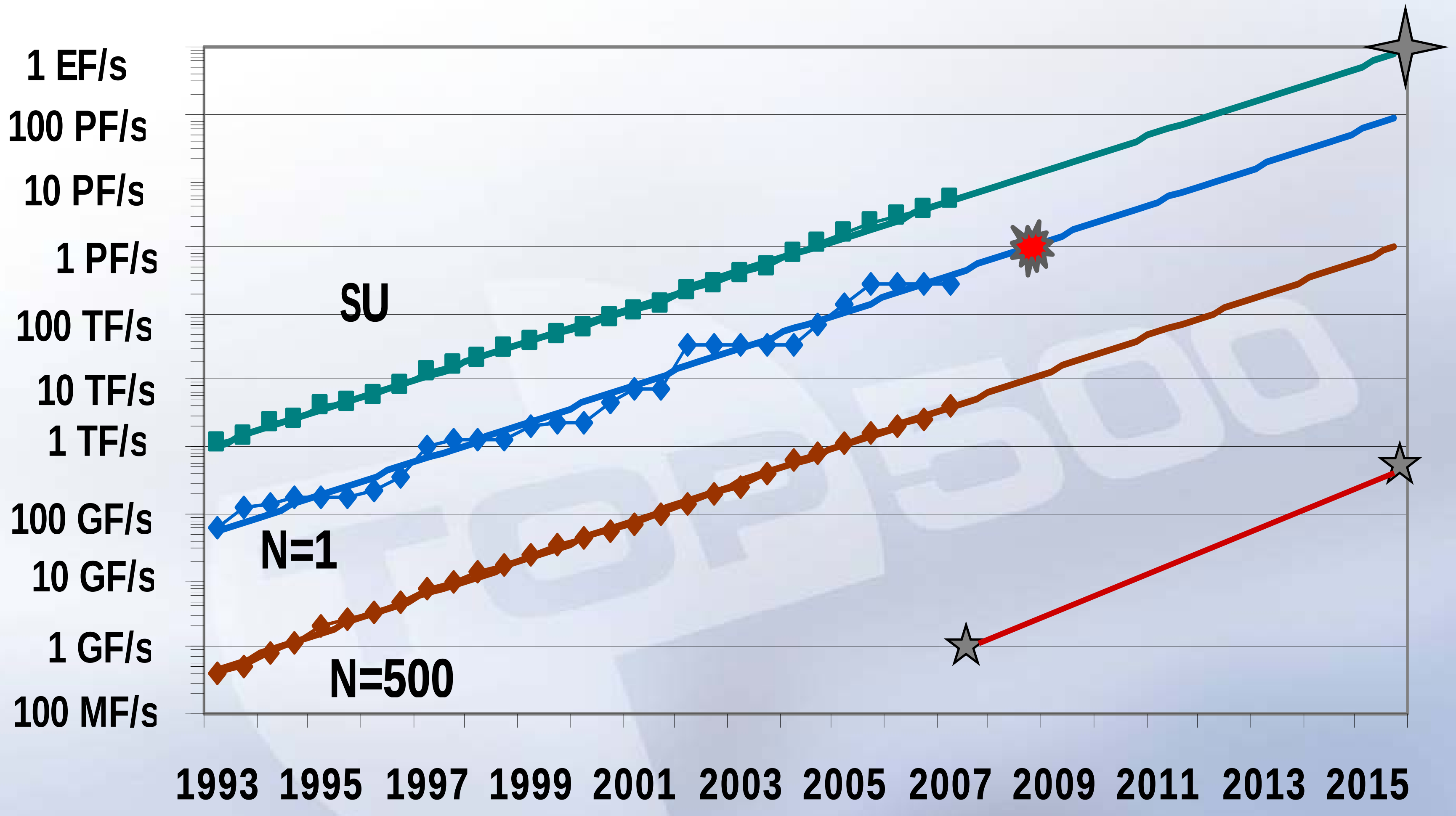
	Manufacturer	Computer	Rmax [TF/s]	Installation Site	Country	Year	#Proc
1	IBM	BlueGene/L eServer Blue Gene	280.6	DOE/NNSA/LLNL	USA	2005	131,072
2 ~~10~~	Cray	Jaguar Cray XT3/XT4	101.7	DOE/ORNL	USA	2007	23,016
3 ~~2~~	Sandia/Cray	Red Storm Cray XT3	101.4	DOE/NNSA/Sandia	USA	2006	26,544
4 ~~3~~	IBM	BGW eServer Blue Gene	91.29	IBM Thomas Watson	USA	2005	40,960
5	IBM	New York BLue eServer Blue Gene	82.16	Stony Brook/BNL	USA	2007	36,864
6 ~~4~~	IBM	ASC Purple eServer pSeries p575	75.76	DOE/NNSA/LLNL	USA	2005	12,208
7	IBM	BlueGene/L eServer Blue Gene	73.03	Rensselaer Polytechnic Institute/CCNI	USA	2007	32,768
8	Dell	Abe PowerEdge 1955, Infiniband	62.68	NCSA	USA	2007	9,600
9 ~~5~~	IBM	MareNostrum JS21 Cluster, Myrinet	62.63	Barcelona Supercomputing Center	Spain	2006	12,240
10	SGI	HLRB-II SGI Altix 4700	56.52	LRZ	Germany	2007	9,728

Countries / Systems (June 2007)

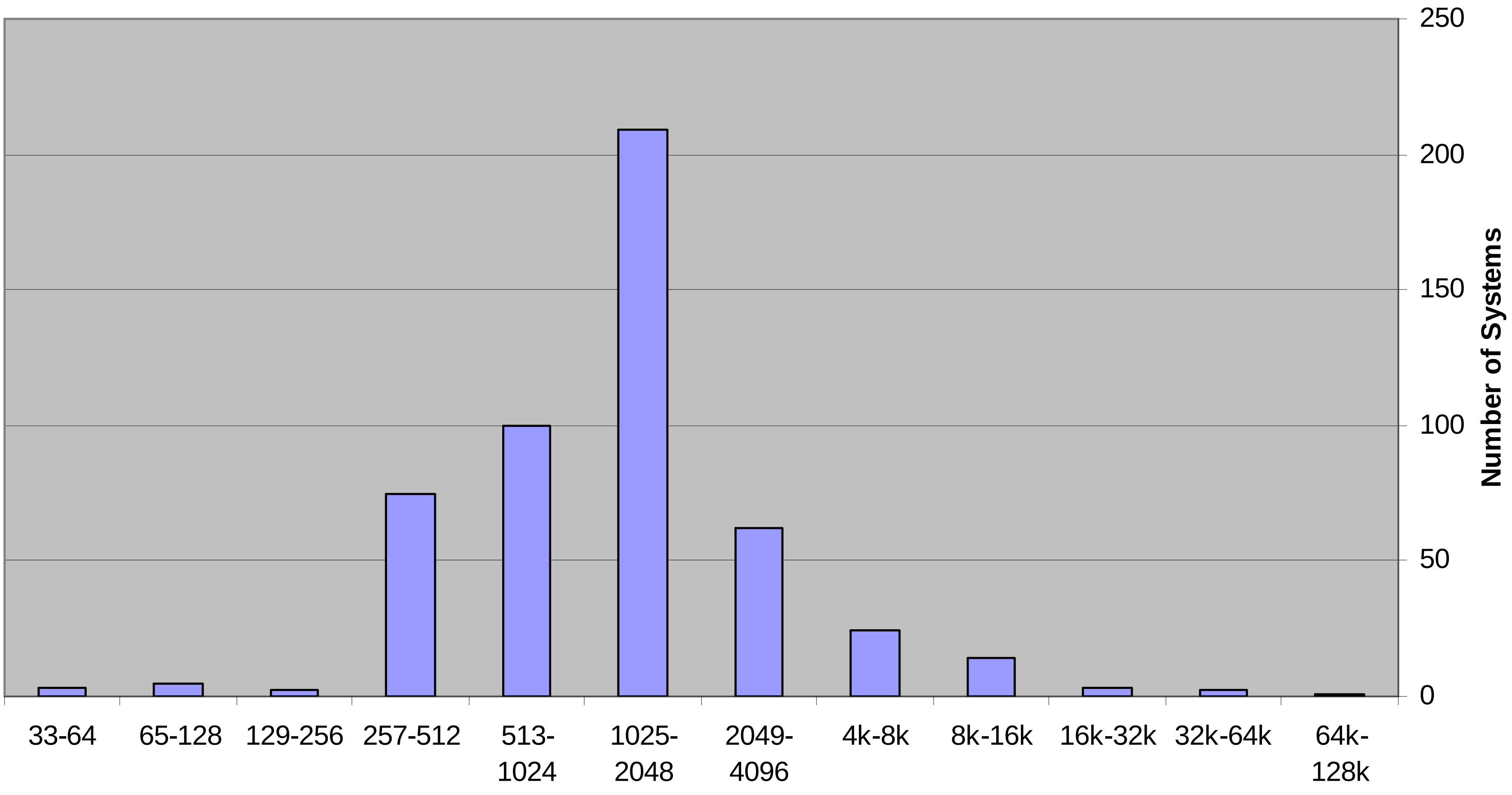
June 2007



Performance Projection

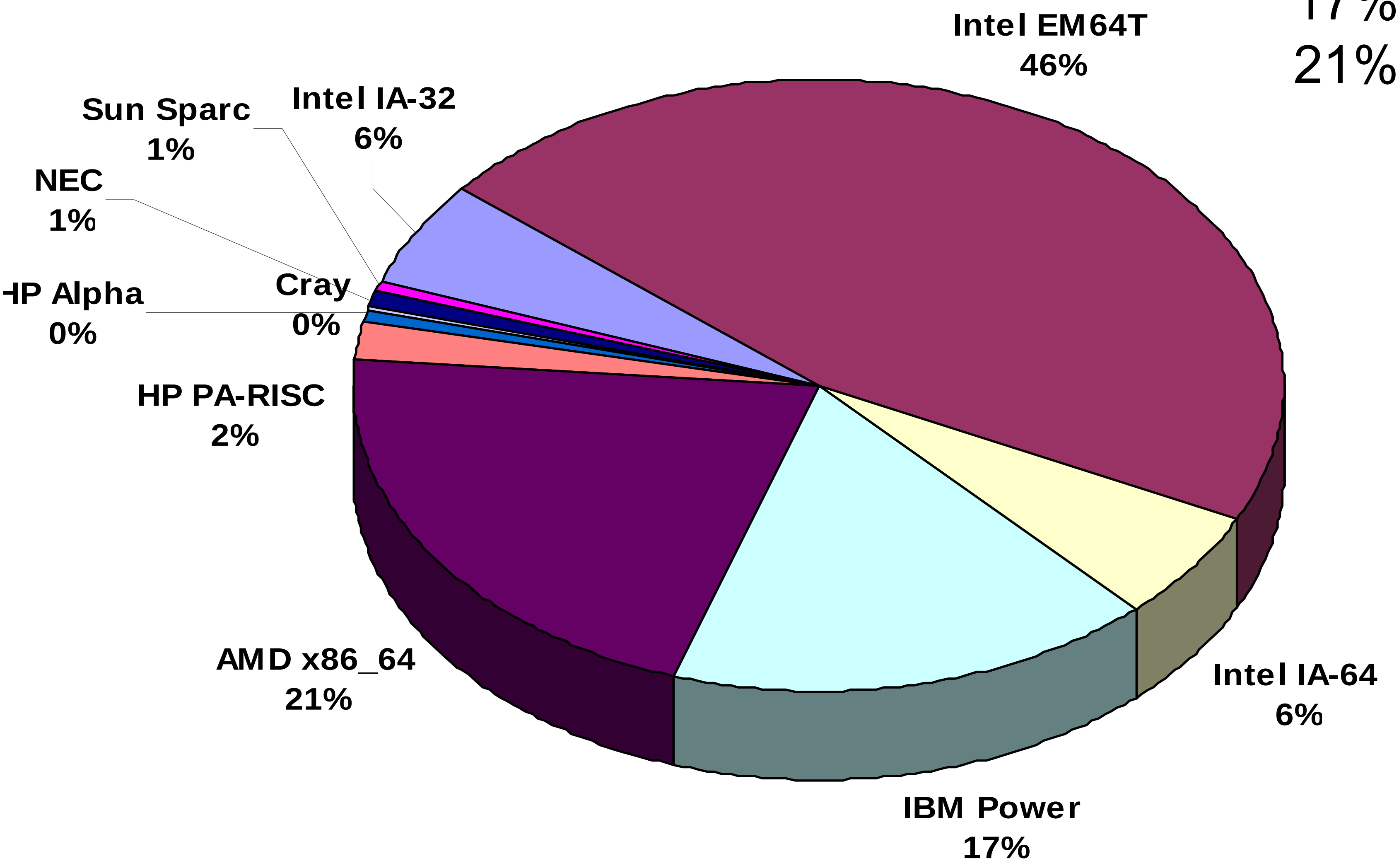


Cores per System – June 2007

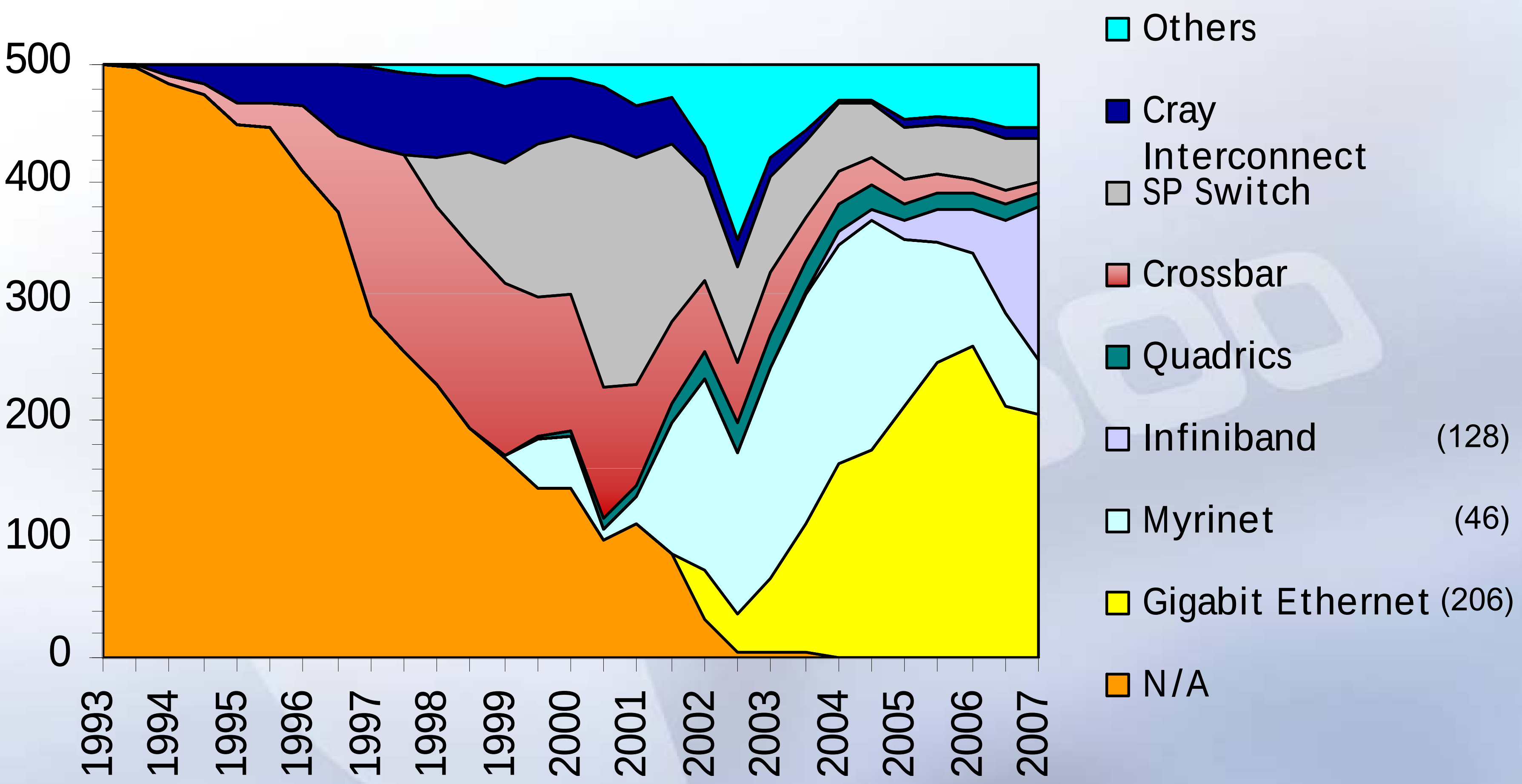


Chips Used in Each of the 500 Systems

96% = 58% Intel
17% IBM
21% AMD



Interconnects / Systems

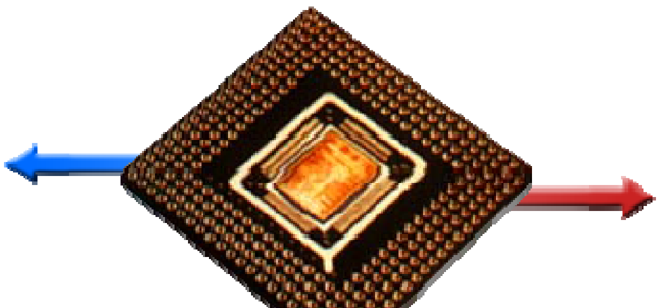


GigE + Infiniband + Myrinet = 76%

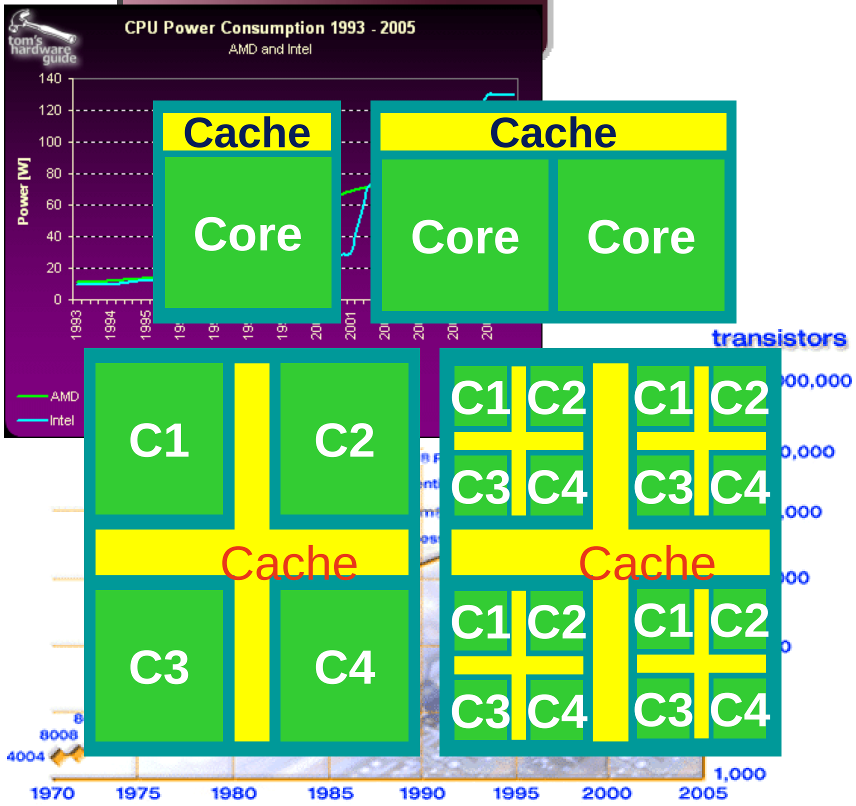
Increasing CPU Performance: A Delicate Balancing Act

Increasing the number of gates into a tight knot and decreasing the cycle time of the processor

Lower Voltage



Increase Clock Rate & Transistor Density



We have seen increasing number of gates on a chip and increasing clock speed.

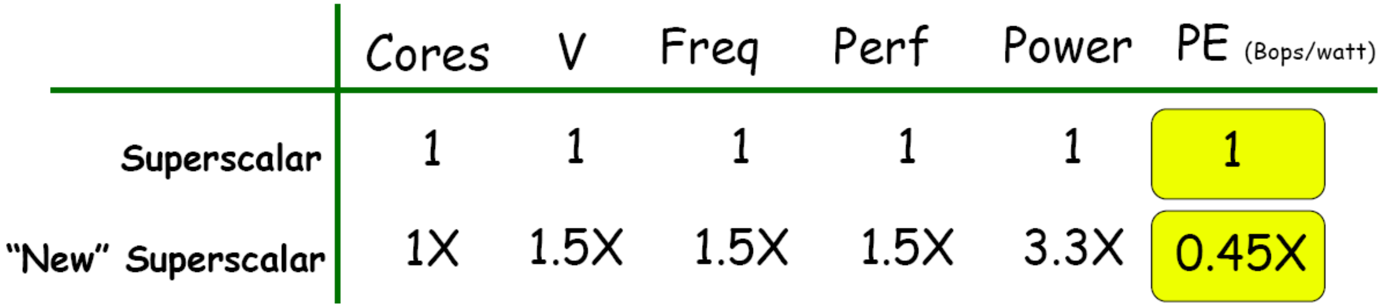
Heat becoming an unmanageable problem, Intel Processors > 100 Watts

We will not see the dramatic increases in clock speeds in the future.

However, the number of gates on a chip will continue to increase.

Power Cost of Frequency

- Power ∝ Voltage^2 x Frequency (V^{2}F)
- **Frequency ∝ Voltage**
- Power ∝Frequency^3



	Cores	V	Freq	Perf	Power	PE (Bops/watt)
Superscalar	1	1	1	1	1	1
"New" Superscalar	1X	1.5X	1.5X	1.5X	3.3X	0.45X

Power Cost of Frequency

- Power ∝ Voltage^2 x Frequency (V^{2}F)
- **Frequency ∝ Voltage**
- Power ∝Frequency^3

	Cores	V	Freq	Perf	Power	PE (Bops/watt)
Superscalar	1	1	1	1	1	1
"New" Superscalar	1X	1.5X	1.5X	1.5X	3.3X	0.45X
Multicore	2X	0.75X	0.75X	1.5X	0.8X	1.88X

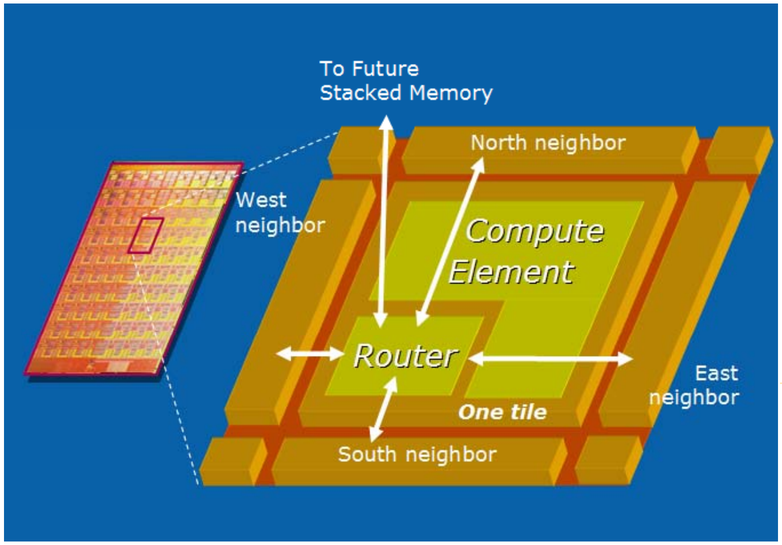
(Bigger # is better)

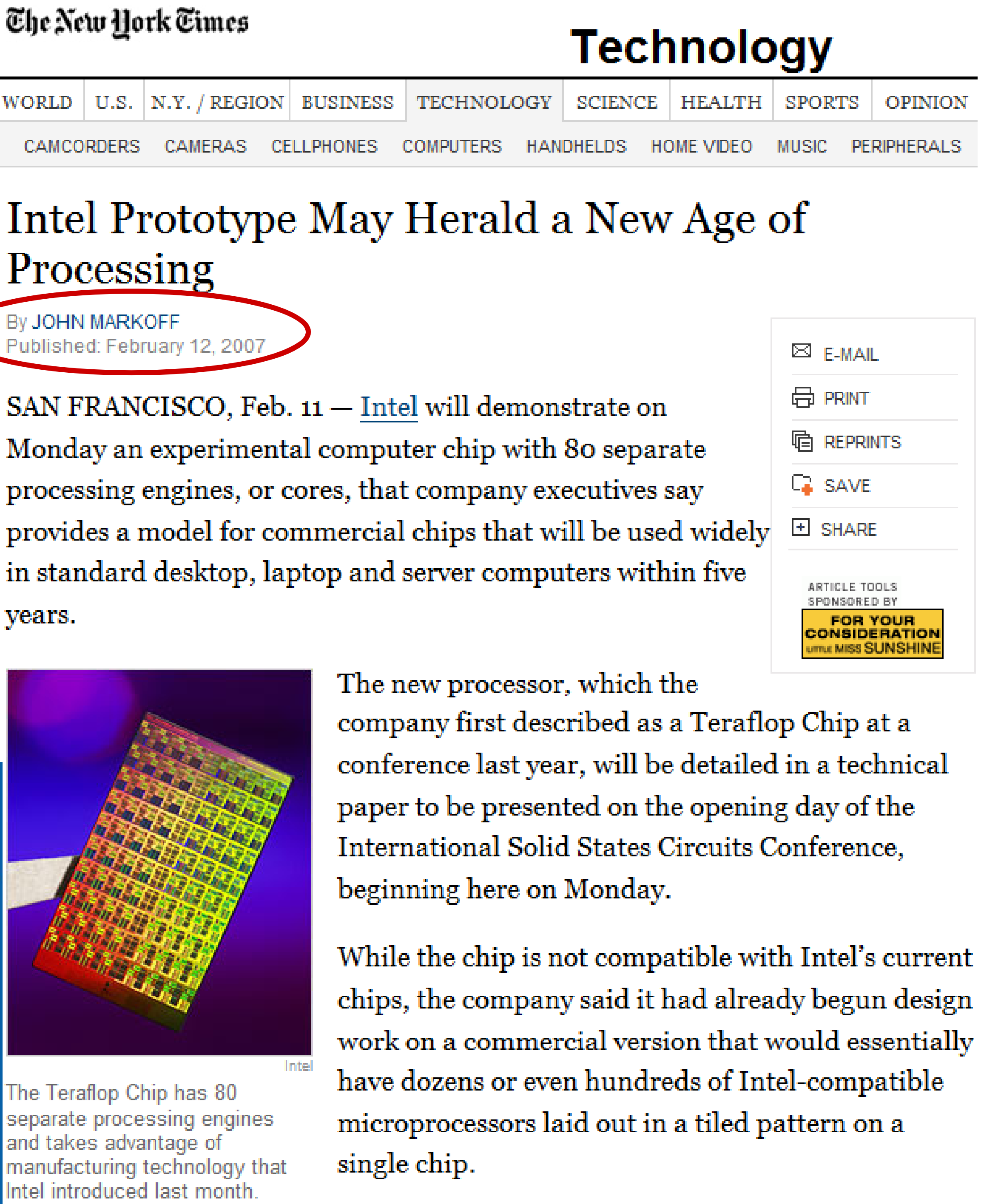
50% more performance with 20% less power

Preferable to use multiple slower devices, than one superfast device

80 Core

- **Intel's 80 Core chip**
  - **~~1~~2 Tflop/s**
  - **62 Watts**
  - **1.2 TB/s internal BW**





Intel Prototype May Herald a New Age of Processing

By JOHN MARKOFF
Published: February 12, 2007

SAN FRANCISCO, Feb. 11 — Intel will demonstrate on Monday an experimental computer chip with 80 separate processing engines, or cores, that company executives say provides a model for commercial chips that will be used widely in standard desktop, laptop and server computers within five years.

Intel

The Teraflop Chip has 80 separate processing engines and takes advantage of manufacturing technology that Intel introduced last month.

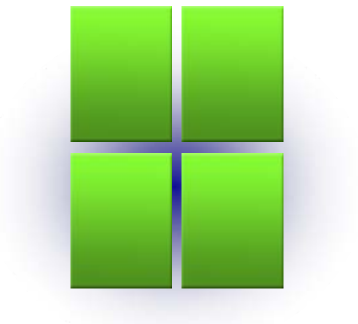
The new processor, which the company first described as a Teraflop Chip at a conference last year, will be detailed in a technical paper to be presented on the opening day of the International Solid States Circuits Conference, beginning here on Monday.

While the chip is not compatible with Intel's current chips, the company said it had already begun design work on a commercial version that would essentially have dozens or even hundreds of Intel-compatible microprocessors laid out in a tiled pattern on a single chip.

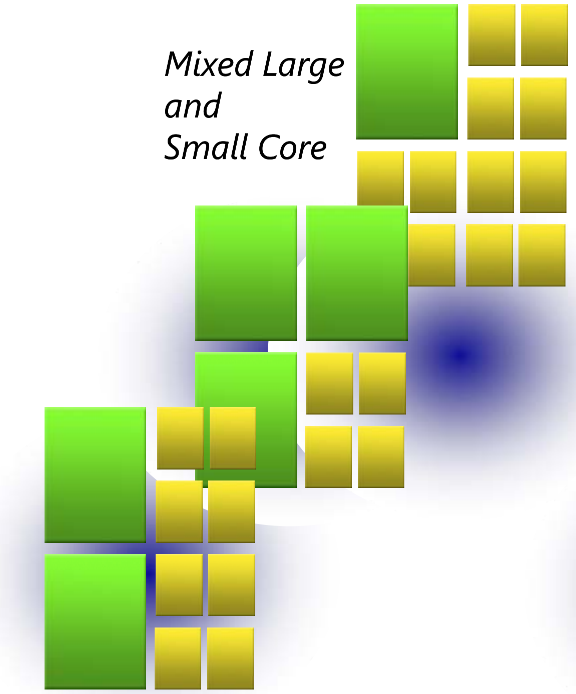
What’s Next?

Different Classes of Chips
- Home
- Games / Graphics
- Business
- Scientific

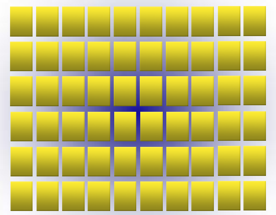
All Large Core



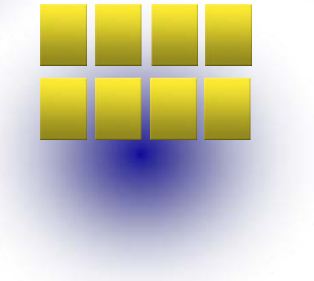
Mixed Large and Small Core



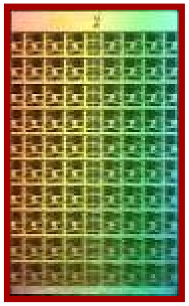
Many Small Cores



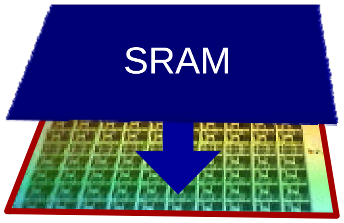
All Small Core



Many Floating-Point Cores



+ 3D Stacked Memory



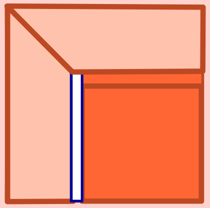
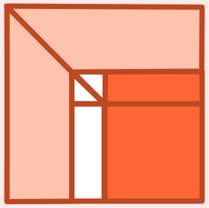
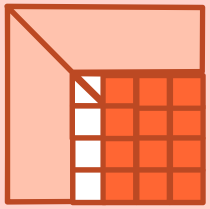
Major Changes to Software

- **Must rethink the design of our software**
  - **Another disruptive technology**
    - **Similar to what happened with cluster computing and message passing**
  - **Rethink and rewrite the applications, algorithms, and software**
- **Numerical libraries for example will change**
  - **For example, both LAPACK and ScaLAPACK will undergo major changes to accommodate this**

Four Important Concepts that Will Effect Math Software

- **Effective Use of Many-Core**
- **Exploiting Mixed Precision in Our Numerical Computations**
- **Self Adapting / Auto Tuning of Software**
- **Fault Tolerant Algorithms**

A New Generation of Software: PLASMA

Algorithms follow hardware evolution along time.		
LINPACK (80's) (Vector operations)		Rely on - Level-1 BLAS operations
LAPACK (90's) (Blocking, cache friendly)		Rely on - Level-3 BLAS operations
PLASMA (00's) New Algorithms (many-core friendly)		Rely on - a DAG/scheduler - block data layout - some extra kernels

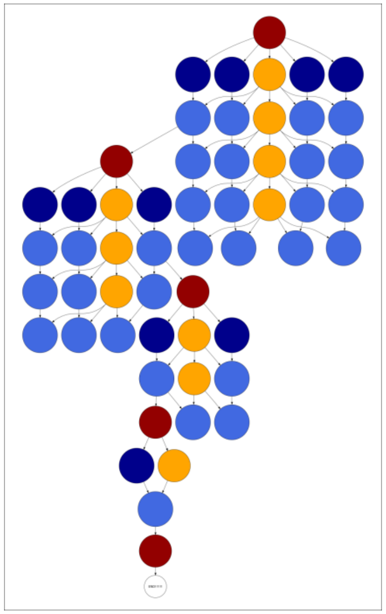
Those new algorithms

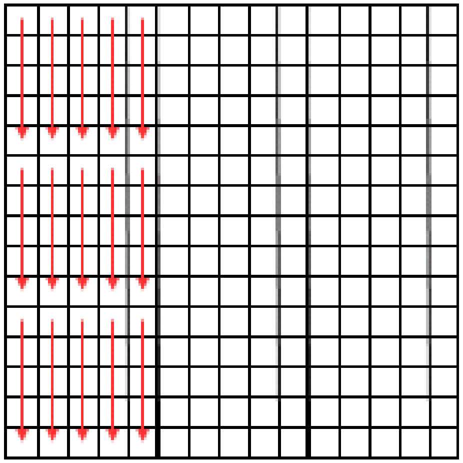
- have a very low granularity, they scale very well (multicore, petascale computing, …)
- removes a lots of dependencies among the tasks, (multicore, distributed computing)
- avoid latency (distributed computing, out-of-core)
- rely on fast kernels

Those new algorithms need new kernels and rely on efficient scheduling algorithms.

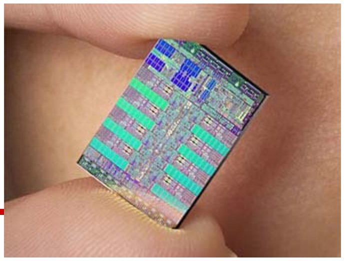
Tuesday Alfredo Buttari’s Talk Track A: 4:20 – 6:00

- **Parallel Tiled QR Factorization for Multicore Architectures**
  - **PLASMA - Parallel Linear Algebra for Scalable Multi-Core Architectures**
    - **Designing the next generation numerical library**

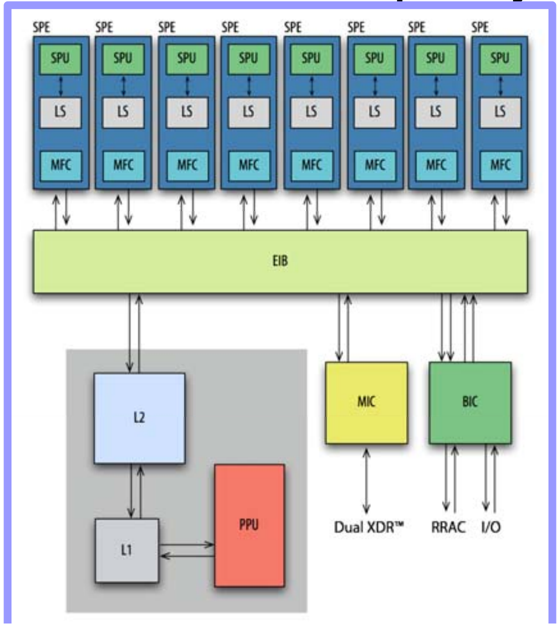




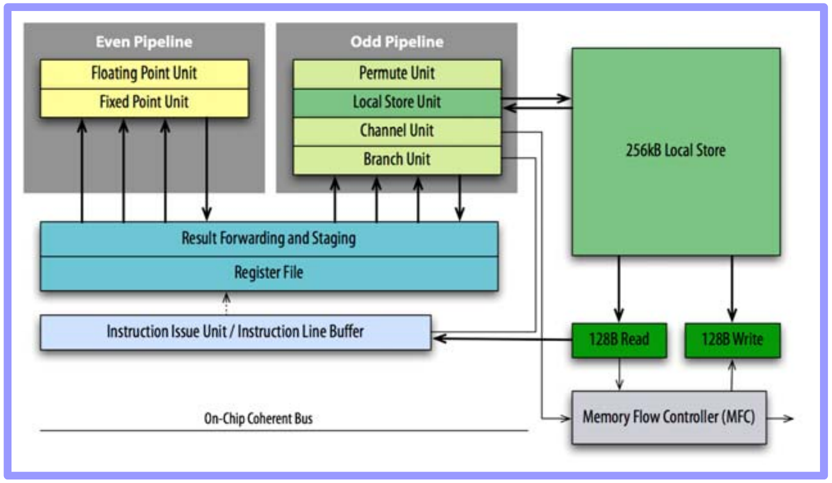
With the Hype on Cell & PS3 We Became Interested



- The PlayStation 3's CPU based on a "Cell“ processor
- Each Cell contains a Power PC processor and 8 SPEs. (SPE is processing unit, SPE: SPU + DMA engine)
  - An SPE is a self contained vector processor which acts independently from the others.
    - 4 way SIMD floating point units capable of a total of 25.6 Gflop/s @ 3.2 GHZ
  - 204.8 Gflop/s peak!
  - The catch is that this is for 32 bit floating point; (Single Precision SP)
  - And 64 bit floating point runs at 14.6 Gflop/s total for all 8 SPEs!!
    - Divide SP peak by 14; factor of 2 because of DP and 7 because of latency issues



SPE ~ 25 Gflop/s peak



Performance of Single Precision on Conventional Processors

- **Realized have the similar situation on our commodity processors.**
  - **That is, SP is 2X as fast as DP on many systems**
- **The Intel Pentium and AMD Opteron have SSE2**
  - **2 flops/cycle DP**
  - **4 flops/cycle SP**
- **IBM PowerPC has AltiVec**
  - **8 flops/cycle SP**
  - **4 flops/cycle DP**
    - **No DP on AltiVec**

	Size	**SGEMM/ DGEMM**	**Size**	**SGEMV/ DGEMV**
AMD Opteron 246	3000	2.00	5000	1.70
UltraSparc-IIe	3000	1.64	5000	1.66
Intel PIII Coppermine	3000	2.03	5000	2.09
PowerPC 970	3000	2.04	5000	1.44
Intel Woodcrest	3000	1.81	5000	2.18
Intel XEON	3000	2.04	5000	1.82
Intel Centrino Duo	3000	2.71	5000	2.21

Single precision is faster because:

- Higher parallelism in SSE/vector units
- Reduced data motion
- Higher locality in cache

32 or 64 bit Floating Point Precision?

- **A long time ago 32 bit floating point was used**
  - **Still used in scientific apps but limited**
- **Most apps use 64 bit floating point**
  - **Accumulation of round off error**
    - **A 10 TFlop/s computer running for 4 hours performs > 1 Exaflop (10^{18}) ops.**
  - **Ill conditioned problems**
  - **IEEE SP exponent bits too few (8 bits, $10^{\pm 38}$)**
  - **Critical sections need higher precision**
    - **Sometimes need extended precision (128 bit fl pt)**
  - **However some can get by with 32 bit fl pt in some parts**
- **Mixed precision a possibility**
  - **Approximate in lower precision and then refine or improve solution to high precision.**

Idea Goes Something Like This…

- **Exploit 32 bit floating point as much as possible.**
  - **Especially for the bulk of the computation**
- **Correct or update the solution with selective use of 64 bit floating point to provide a refined results**
- **Intuitively:**
  - **Compute a 32 bit result,**
  - **Calculate a correction to 32 bit result using selected higher precision and,**
  - **Perform the update of the 32 bit results with the correction using high precision.**

Mixed-Precision Iterative Refinement

- **Iterative refinement for dense systems,** $Ax = b$**, can work this way.**

```
L U = lu(A)                   SINGLE       $O(n^3)$
x = L\(U\b)                   SINGLE       $O(n^2)$
r = b – Ax                    DOUBLE       $O(n^2)$
WHILE || r || not small enough
    z = L\(U\r)               SINGLE       $O(n^2)$
    x = x + z                 DOUBLE       $O(n^1)$
    r = b – Ax                DOUBLE       $O(n^2)$
END
```

Mixed-Precision Iterative Refinement

- **Iterative refinement for dense systems,** $Ax = b$, **can work this way.**

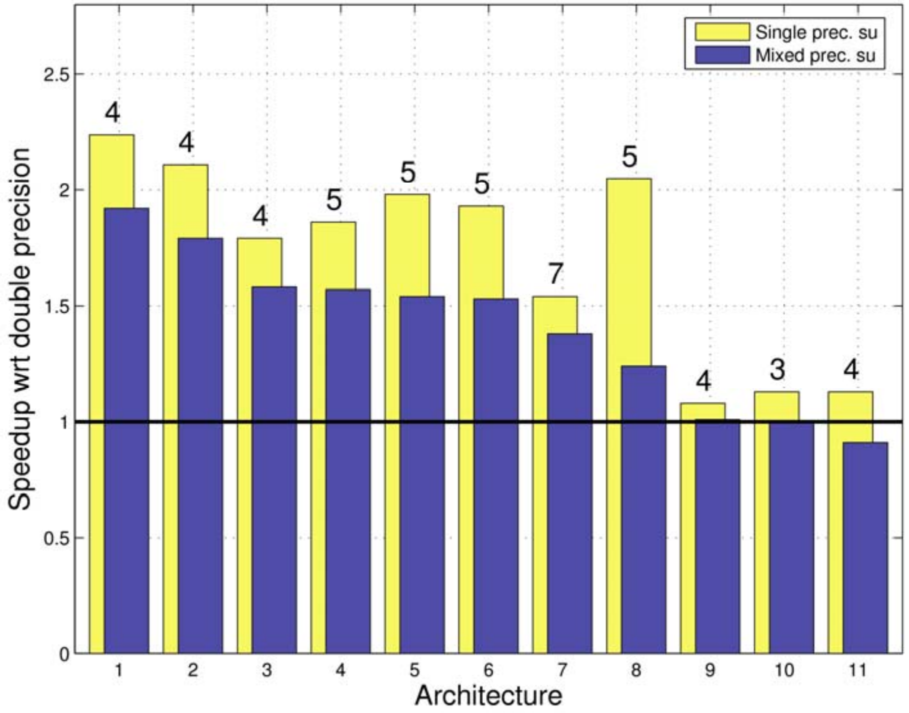
```
L U = lu(A)                   SINGLE      $O(n^3)$
x = L\(U\b)                   SINGLE      $O(n^2)$
r = b – Ax                    DOUBLE      $O(n^2)$
WHILE || r || not small enough
    z = L\(U\r)               SINGLE      $O(n^2)$
    x = x + z                 DOUBLE      $O(n^1)$
    r = b – Ax                DOUBLE      $O(n^2)$
END
```

- **Wilkinson, Moler, Stewart, & Higham provide error bound for SP fl pt results when using DP fl pt.**
- **It can be shown that using this approach we can compute the solution to 64-bit floating point precision.**

- **Requires extra storage, total is 1.5 times normal;**
- **$O(n^3)$ work is done in lower precision**
- **$O(n^2)$ work is done in high precision**

- **Problems if the matrix is ill-conditioned in sp; $O(10^8)$**

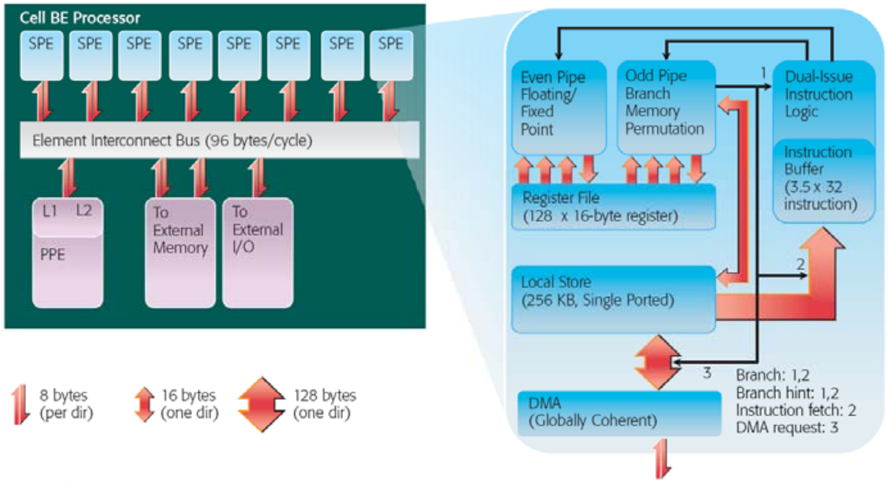
Results for Multiple Precision Iterative Refinement



	Architecture (BLAS)
1	Intel Pentium III Coppermine (Goto)
2	Intel Pentium III Katmai (Goto)
3	Sun UltraSPARC IIe (Sunperf)
4	Intel Pentium IV Prescott (Goto)
5	Intel Pentium IV-M Northwood (Goto)
6	AMD Opteron (Goto)
7	Cray X1 (libsci)
8	IBM Power PC G5 (2.7 GHz) (VecLib)
9	Compaq Alpha EV6 (CXML)
10	IBM SP Power3 (ESSL)
11	SGI Octane (ATLAS)

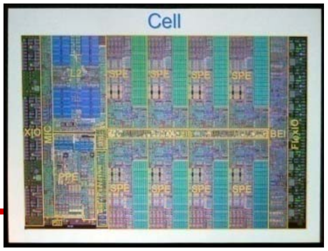
New routines in LAPACK that do this for LU and LL^T

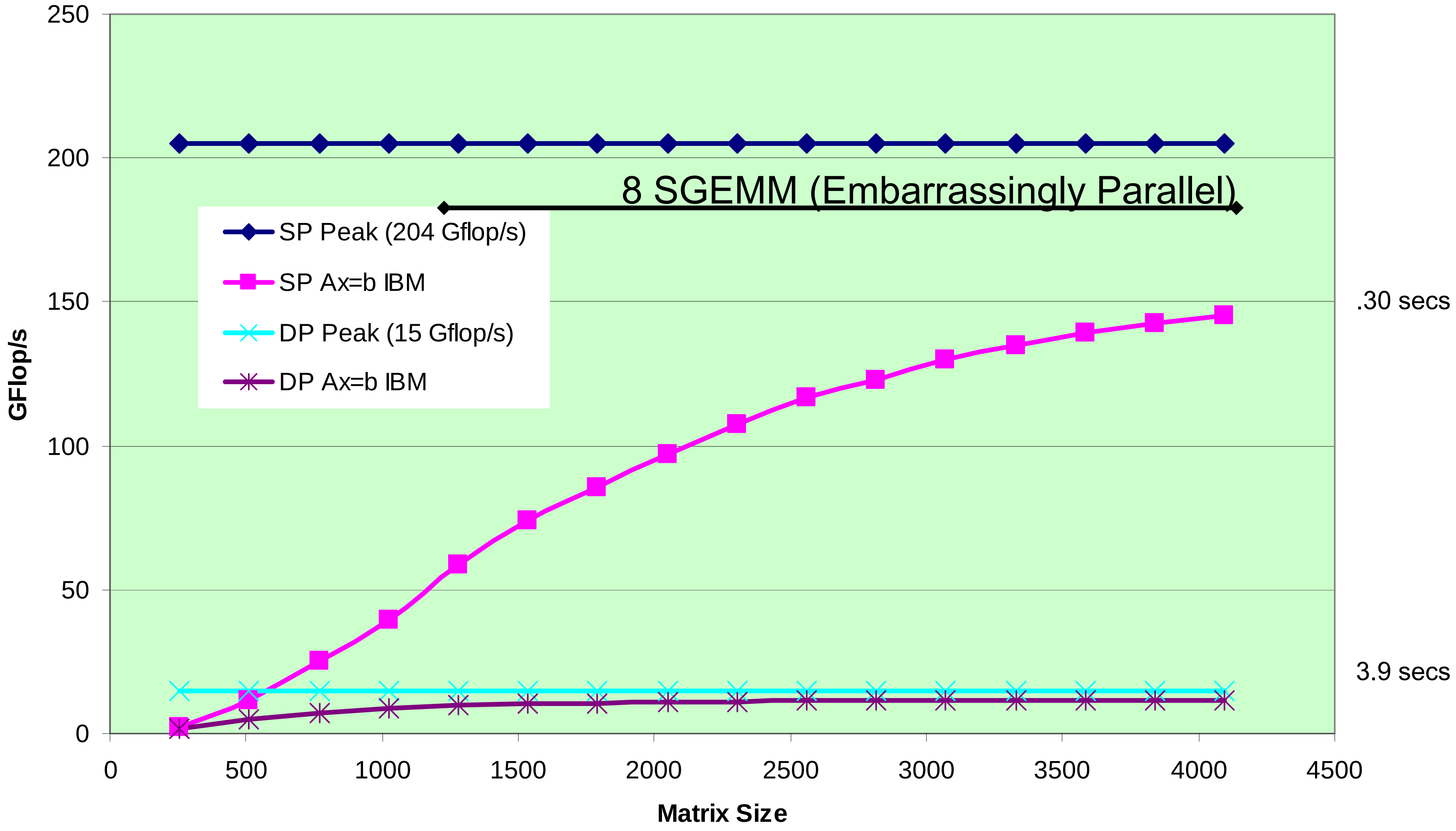
What about the Cell?


Cell BE Processor
SPE
SPE
SPE
SPE
SPE
SPE
SPE
SPE
Element Interconnect Bus (96 bytes/cycle)
L1
L2
PPE
To External Memory
To External I/O
8 bytes (per dir)
16 bytes (one dir)
128 bytes (one dir)
Even Pipe Floating/ Fixed Point
Odd Pipe Branch Memory Permutation
1
Dual-Issue Instruction Logic
Instruction Buffer (3.5 x 32 instruction)
Register File (128 x 16-byte register)
2
Local Store (256 KB, Single Ported)
3
DMA (Globally Coherent)
Branch: 1,2
Branch hint: 1,2
Instruction fetch: 2
DMA request: 3

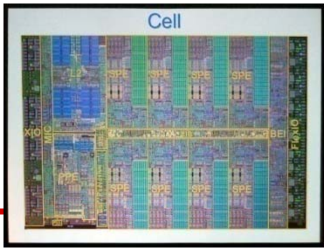
- **Power PC at 3.2 GHz**
  - **DGEMM at 5 Gflop/s**
  - **Altivec peak at 25.6**
    - **Achieved 10 Gflop/s SGEMM**
- **8 SPUs**
  - **204.8 Gflop/s peak!**
  - **The catch is that this is for 32 bit floating point; (Single Precision SP)**
  - **And 64 bit floating point runs at 14.6 Gflop/s total for all 8 SPEs!!**
    - **Divide SP peak by 14; factor of 2 because of DP and 7 because of latency issues**

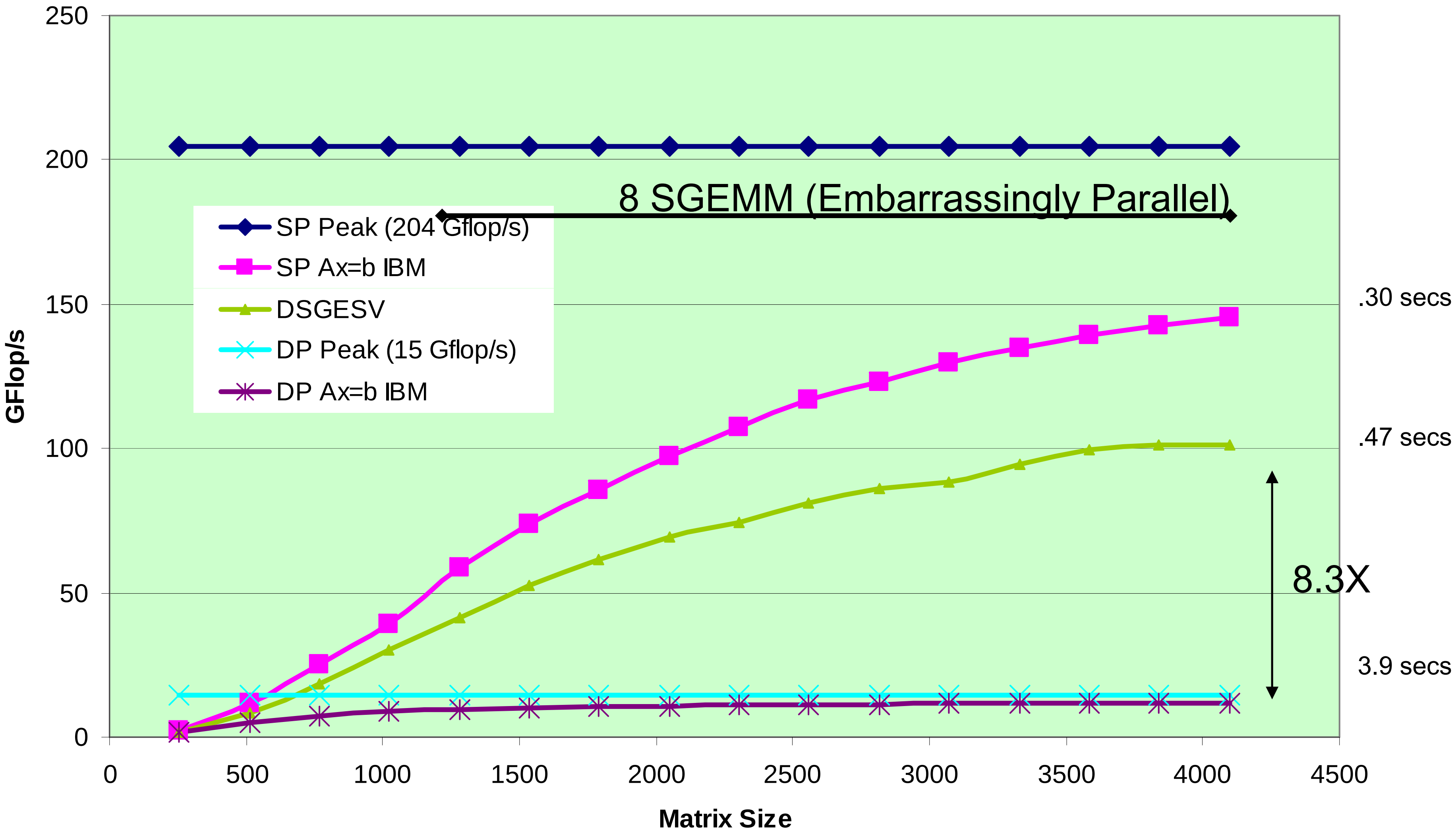
IBM Cell 3.2 GHz, Ax = b



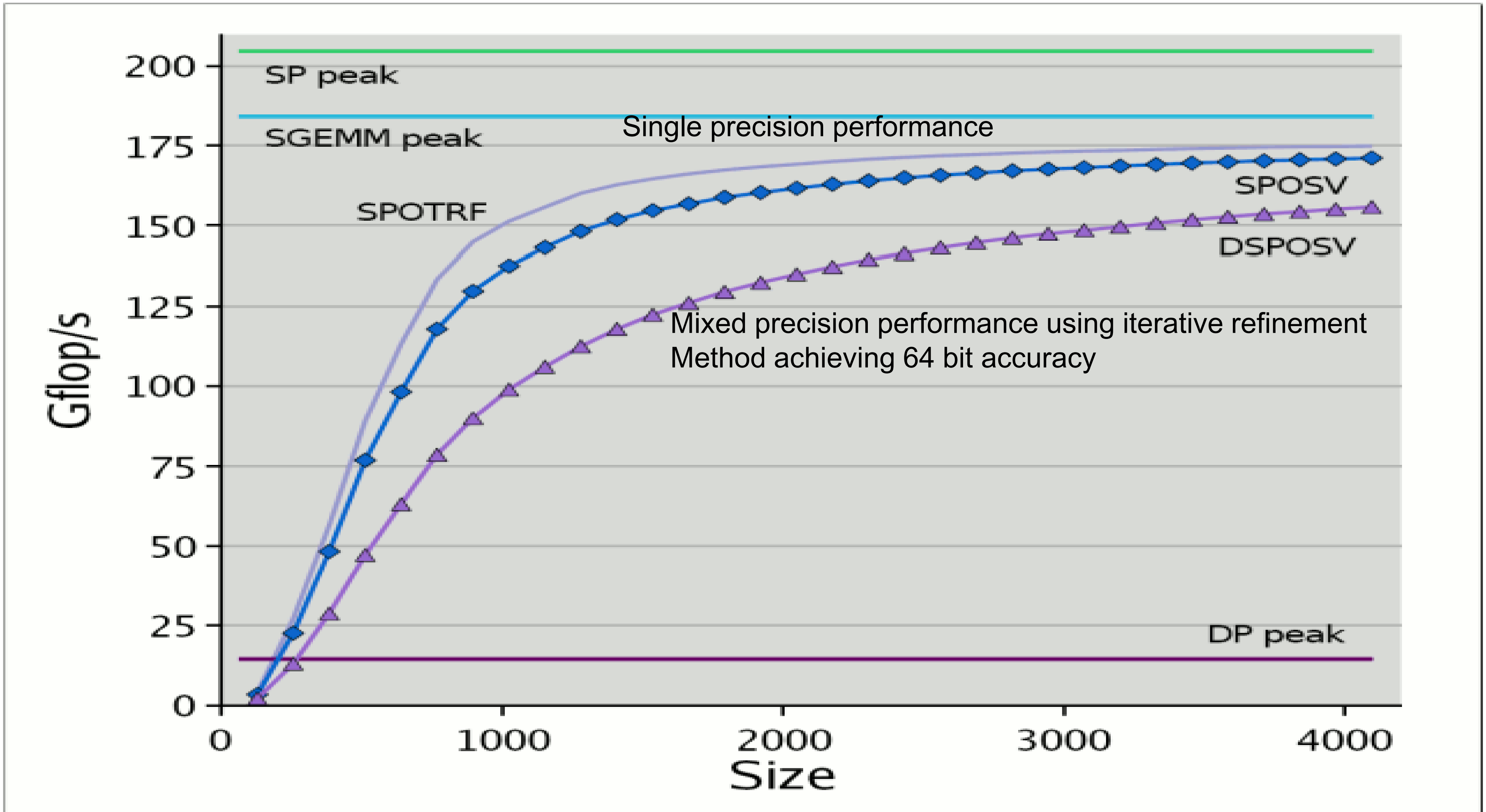


IBM Cell 3.2 GHz, Ax = b





Cholesky on the Cell, $Ax=b$, $A=A^T$, $x^TAx > 0$



Single precision performance

Mixed precision performance using iterative refinement
Method achieving 64 bit accuracy

For the SPE's standard C code and C language SIMD extensions (intrinsics)

Quadruple Precision

n	Quad Precision Ax = b	Iter. Refine. DP to QP	
	time (s)	time (s)	Speedup
100	0.29	0.03	9.5
200	2.27	0.10	20.9
300	7.61	0.24	30.5
400	17.8	0.44	40.4
500	34.7	0.69	49.7
600	60.1	1.01	59.0
700	94.9	1.38	68.7
800	141.	1.83	77.3
900	201.	2.33	86.3
1000	276.	2.92	94.8

Intel Xeon 3.2 GHz

Reference implementation of the quad precision BLAS

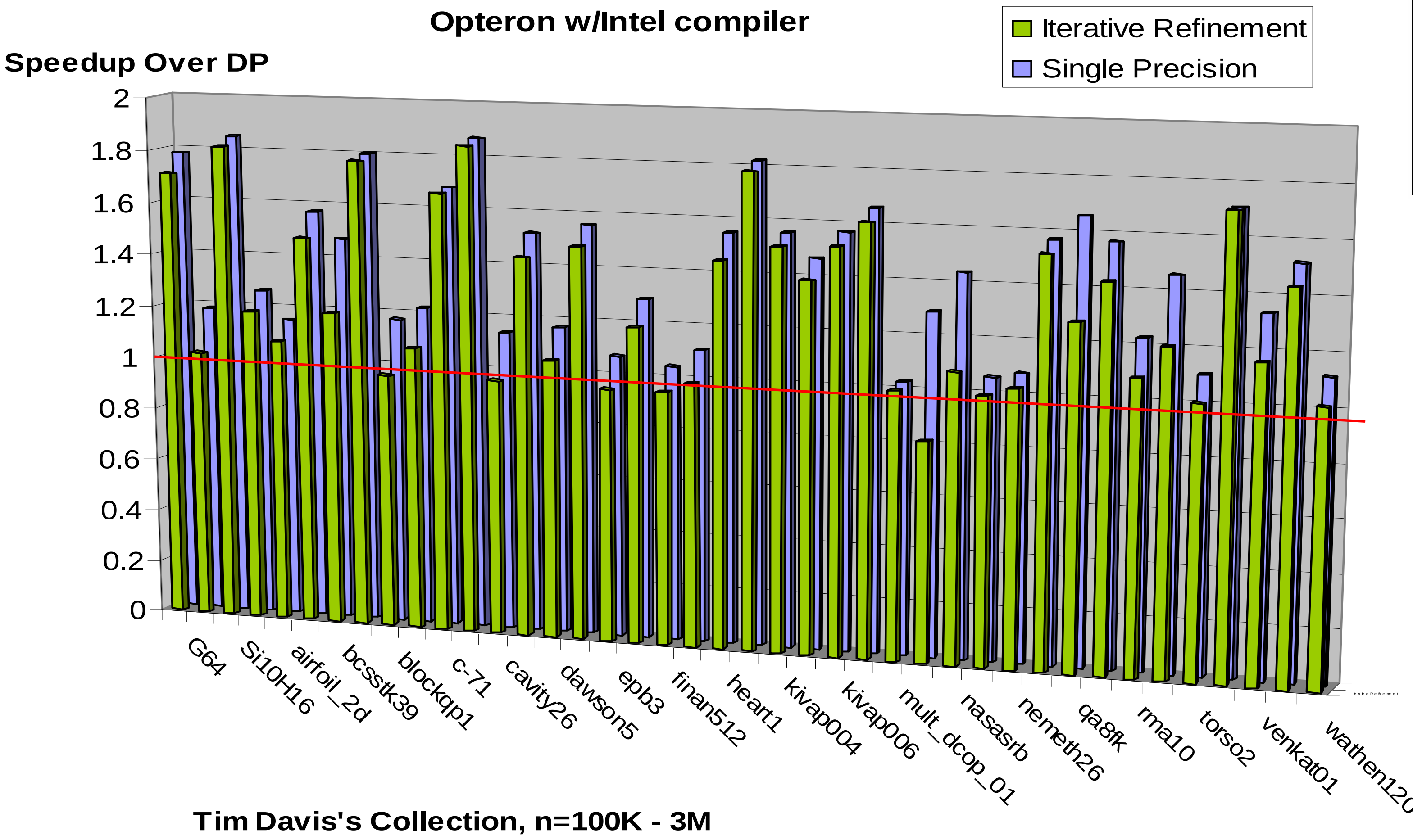
Accuracy: 10^{-32}

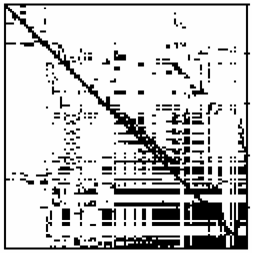
No more than 3 steps of iterative refinement are needed.

- **Variable precision factorization (with say < 32 bit precision) plus 64 bit refinement produces 64 bit accuracy**

Sparse Direct Solver and Iterative Refinement

MUMPS package based on multifrontal approach which generates small dense matrix multiplies





Sparse Iterative Methods (PCG)

- **Outer/Inner Iteration**

Outer iterations using 64 bit floating point

Inner iteration:
In 32 bit floating point

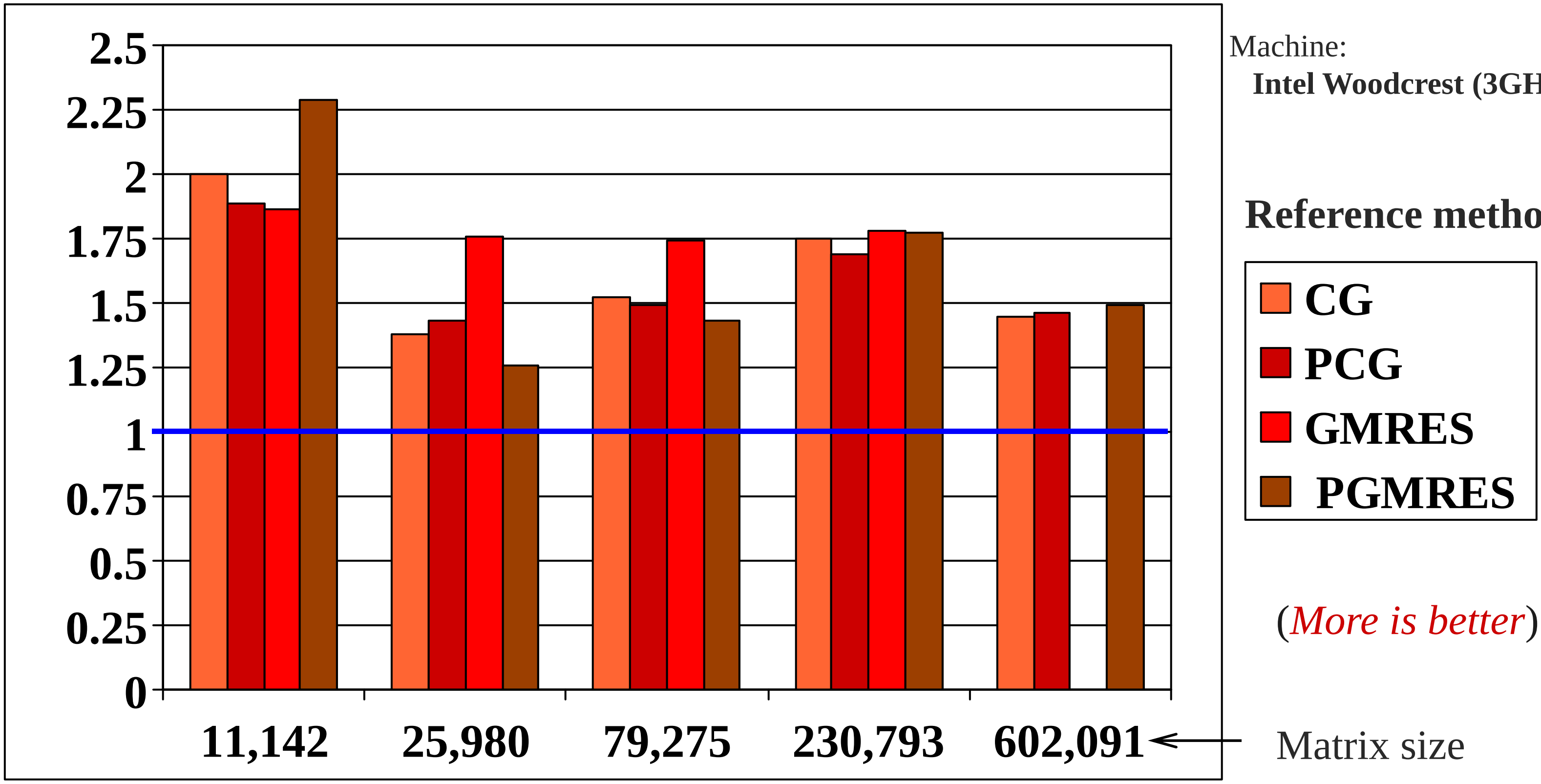
```
Compute $r^{(0)} = b - Ax^{(0)}$ for some initial guess $x^{(0)}$
for  $i = 1, 2, \ldots$
    solve $Mz^{(i-1)} = r^{(i-1)}$
    $\rho_{i-1} = r^{(i-1)^T} z^{(i-1)}$
    if $i = 1$
        $p^{(1)} = z^{(0)}$
    else
        $\beta_{i-1} = \rho_{i-1}/\rho_{i-2}$
        $p^{(i)} = z^{(i-1)} + \beta_{i-1} p^{(i-1)}$
    endif
    $q^{(i)} = Ap^{(i)}$
    $\alpha_i = \rho_{i-1}/p^{(i)^T} q^{(i)}$
    $x^{(i)} = x^{(i-1)} + \alpha_i p^{(i)}$
    $r^{(i)} = r^{(i-1)} - \alpha_i q^{(i)}$
    check convergence; continue if necessary
end
```

```
Compute $r^{(0)} = b - Ax^{(0)}$ for some initial guess $x^{(0)}$
for  $i = 1, 2, \ldots$
    solve $Mz^{(i-1)} = r^{(i-1)}$
    $\rho_{i-1} = r^{(i-1)^T} z^{(i-1)}$
    if $i = 1$
        $p^{(1)} = z^{(0)}$
    else
        $\beta_{i-1} = \rho_{i-1}/\rho_{i-2}$
        $p^{(i)} = z^{(i-1)} + \beta_{i-1} p^{(i-1)}$
    endif
    $q^{(i)} = Ap^{(i)}$
    $\alpha_i = \rho_{i-1}/p^{(i)^T} q^{(i)}$
    $x^{(i)} = x^{(i-1)} + \alpha_i p^{(i)}$
    $r^{(i)} = r^{(i-1)} - \alpha_i q^{(i)}$
    check convergence; continue if necessary
end
```

- **Outer iteration in 64 bit floating point and fixed number of inner iteration in 32 bit floating point**

Mixed Precision Computations for Sparse Inner/Outer-type Iterative Solvers

Time speedups for mixed precision Inner SP/Outer DP (SP/DP) iter. methods *vs* DP/DP
(CG, GMRES, PCG, and PGMRES with diagonal preconditioners)



Machine:
Intel Woodcrest (3GHz, 1333MHz bus)

(*More is better*)

← Matrix size

6,021 18,000 39,000 120,000 240,000 ← Condition number

Intriguing Potential

- **Exploit lower precision as much as possible**
  - **Payoff in performance**
    - **Faster floating point**
    - **Less data to move**
- **Automatically switch between SP and DP to match the desired accuracy**
  - **Compute solution in SP and then a correction to the solution in DP**
- **Potential for GPU, FPGA, special purpose processors**
  - **What about 16 bit floating point?**
    - **Use as little you can get away with and improve the accuracy**
- **Linear systems and Eigenvalue, optimization problems, where Newton's method is used.**

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

$$\boxed{x_{i+1} - x_i} = -\frac{f(x_i)}{f'(x_i)}$$

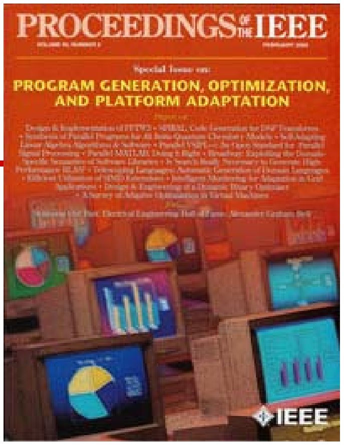
Correction = - A\(b – Ax)

How to Deal with Complexity?

- **Complexity is increasing in our systems and efficiency of our software is going down.**
  - **More parallelism, hardware complexity**
- **Handwritten code is**
  - **Increasing difficult to develop**
  - **Expensive**
  - **Rapidly outdated**
- **Adaptivity is the key for applications to effectively use available resources whose complexity is exponentially increasing**
- **Goal:**
  - **Automatically bridge the gap between the application and computers that are rapidly changing and getting more and more complex**

Examples of Automatic Performance Tuning

Proceedings of the IEEE,
V: 93 #: 2 Feb. 2005
Issue on Program
Generation,
Optimization, and
Platform Adaptation



PROCEEDINGS OF THE IEEE

Special Issue on:

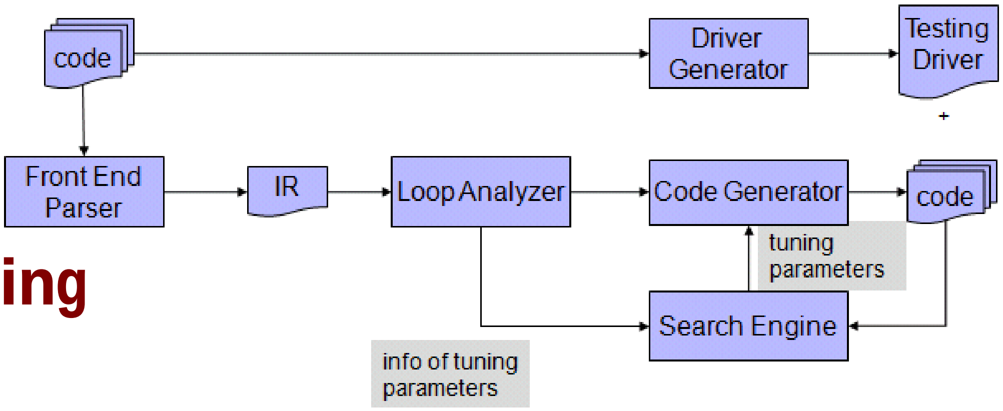
PROGRAM GENERATION, OPTIMIZATION, AND PLATFORM ADAPTATION

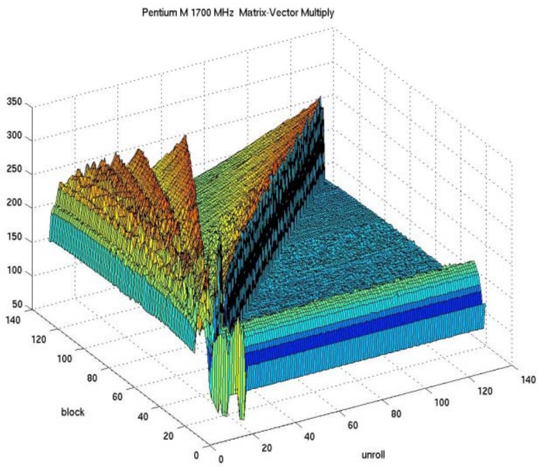
IEEE

- Dense BLAS
  - Sequential
  - ATLAS (UTK) & PHiPAC (UCB)
- Fast Fourier Transform (FFT) & variations
  - FFTW (MIT)
  - Sequential and Parallel
  - www.fftw.org
- Digital Signal Processing
  - SPIRAL: www.spiral.net (CMU)
- MPI Collectives (UCB, UTK)
- More projects, conferences, government reports, …

Generic Code Optimization

- **Can ATLAS-like techniques be applied to arbitrary code?**
- **What do we mean by ATLAS-like techniques?**
  - **Blocking**
  - **Loop unrolling**
  - **Data prefetch**
  - **Functional unit scheduling**
  - **etc.**
- **Referred to as** ***empirical optimization***
  - **Generate many variations**
  - **Pick the best implementation by measuring the performance**



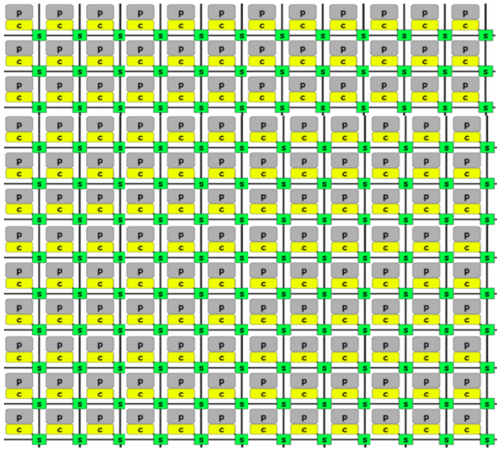


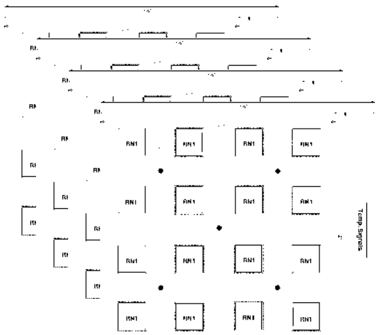
Applying Self Adapting Software

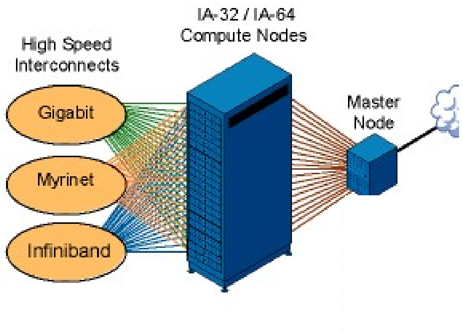
- **Numerical and Non-numerical applications**
  - **BLAS like ops / message passing collectives**
- **Static or Dynamic determine code to be used**
  - **Perform at make time / every time invoked**
- **Independent or dependent on data presented**
  - **Same on each data set / depends on properties of data**

Future Large Systems, Say in a Few Years

- **128 cores per socket**
- **32 sockets per node**
- **128 nodes per system**
- **System = 128*32*128**
  = 524,288 Cores!
- **And by the way, its 4 threads of exec per core**
- **That's about 2M threads to manage**







Conclusions

- **For the last decade or more, the research investment strategy has been overwhelmingly biased in favor of hardware.**
- **This strategy needs to be rebalanced - barriers to progress are increasingly on the software side.**
- **Moreover, the return on investment is more favorable to software.**
  - **Hardware has a half-life measured in years, while software has a half-life measured in decades.**
- **High Performance Ecosystem out of balance**
  - **Hardware, OS, Compilers, Software, Algorithms, Applications**
    - **No Moore's Law for software, algorithms and applications**

Collaborators / Support

Alfredo Buttari, UTK

Julien Langou, UColorado

Julie Langou, UTK

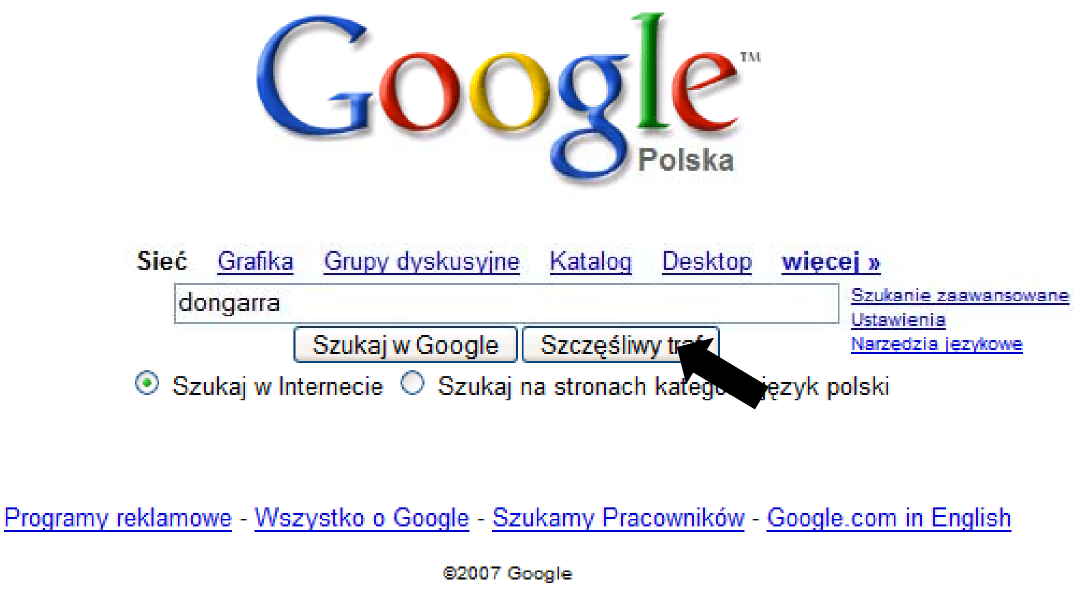
Piotr Luszczek, MathWorks

Jakub Kurzak, UTK

Stan Tomov, UTK

Microsoft

The MathWorks



Sieć Grafika Grupy dyskusyjne Katalog Desktop więcej »

dongarra

Szukaj w Google
Szczęśliwy tr
Szukanie zaawansowane
Ustawienia
Narzędzia językowe

● Szukaj w Internecie ○ Szukaj na stronach katego język polski

Programy reklamowe - Wszystko o Google - Szukamy Pracowników - Google.com in English

©2007 Google